CONNECTED AND OVERLAPPED SHAPES ENHANCEMENTS

Cross-Reference to Related Application

[0001] The present application claims the benefit of U.S. Provisional Patent Application No. 60/445,625 filed on February 7, 2003, which is hereby incorporated by reference.

Field of the Invention

[0002] The present invention relates generally to diagramming and more particularly to methods of creating and manipulating diagrams and to diagramming tools for creating and manipulating diagrams.

Background of the Invention

[0003] Computer software programs that allow a user to create diagrams including linked visual objects are well known in the art. To-date, these computer software programs have been limited, allowing users to create diagrams that only include visual objects of simple shape, linked with lines and laid out in a two-dimensional plane without overlap. The simple shapes of visual objects available to users have typically included rectangles, ellipses and simple polygons. In many instances, these computer software programs statically attach arrows between linked visual objects or draw lines between linked visual objects without arrows. Although these computer software programs have proven to be useful, a need exists for diagramming tools that enable a user to create enhanced diagrams.

[0004] It is therefore an object of the present invention to provide novel methods of creating and manipulating diagrams and diagramming tools for creating and manipulating diagrams.

Summary of the Invention

[0005] Accordingly, in one aspect of the present invention there is provided a method of ordering visual objects presented on a display comprising: comparing visual

objects to be placed in an overlapping condition; determining from the comparison the order in which said visual objects are to be placed; and re-ordering and placing the visual objects in said overlapping condition in accordance with said determination.

[0006] In one embodiment, during the comparing, a metric of each visual object is compared to determine a relative size of each visual object. During the determining, visual objects are placed in order from smallest to largest with smaller visual objects being places in front of larger visual objects. The metric may be the area of a rectangular region surrounding each visual object.

[0007] In one embodiment, the comparing, determining and re-ordering are performed automatically when a visual object is brought into an overlapping condition with another visual object during visual object manipulation. In another embodiment, the comparing, determining and re-ordering are performed automatically when a visual object is brought into an overlapping condition with another visual object during visual object manipulation and then released. In still yet another embodiment, the comparing, determining and re-ordering are performed in response to a user input command.

[0008] According to another aspect of the present invention there is provided a method of creating a connecting link between source and destination visual objects comprising: determining a region within each visual object to be joined and a connecting path extending between the regions; clipping the connecting path so that the connecting path terminates at the locations where the connecting path intersects the source and destination visual objects; and extending a connecting link between the source and destination visual objects terminating at said locations.

[0009] In one embodiment, the region is a point within the each visual object and specifically, the center of each visual object. When the connecting path is a straight line, during the clipping, the straight line is traversed to determine the locations where the connecting path intersects the source and destination visual objects. When the connecting path is a curved line, during the clipping, the curved line is flattened and represented by a series of straight line segments. Each straight line segment is traversed to determine the locations where the connecting path intersects the source and destination visual objects. When the connecting path is a self-loop, during the clipping the self-loop is traversed in clockwise and anti-clockwise directions to

determine the locations where the connecting path intersects the source and destination visual objects.

[0010] Arrowheads may be provided at one or both ends of the connecting link. When an arrowhead is placed at the end of a connecting link, the tip of the arrowhead terminates at the location and the connecting link terminates at the backend of the arrowhead.

[0011] The connecting link may be represented by a plurality of space shapes. The spaced shapes are typically generally evenly spaced along the length of the connecting path. The shapes along the connecting path may be the same or different. The shapes provide semantic meaning to the connecting link joining visual objects.

[0012] According to yet another aspect of the present invention there is provided a method of creating a connecting link joining source and destination visual objects comprising: determining a path for said connecting link between said source and destination visual objects; and representing said connecting link by at least one non-line shape.

[0013] According to still yet another aspect of the present invention there is provided an overlapping object tool for ordering visual objects presented on a display comprising: means for comparing visual objects to be placed in an overlapping condition; means for determining the order in which said visual objects are to be placed based on said comparison; and means for re-ordering and placing the visual objects in said overlapping condition in accordance with said determination.

[0014] According to still yet another aspect of the present invention there is provided an object-connecting tool for creating a connecting link between source and destination visual objects comprising: means for determining a region within each visual object to be joined and a connecting path extending between the regions; means for clipping the connecting path so that the connecting path terminates at the locations where the connecting path intersects the source and destination visual objects; and means for extending a connecting link between the source and destination visual objects terminating at said locations.

[0015] According to still yet another aspect of the present invention there is provided an object-connecting tool for creating a connecting link joining source and destination visual objects comprising: means for determining a path for said

connecting link between said source and destination visual objects; and means for representing said connecting link by at least one non-line shape.

[0016] According to still yet another aspect of the present invention there is provided a computer readable medium including a computer program for ordering visual objects presented on a display, said computer program comprising: computer program code for comparing visual objects to be placed in an overlapping condition; computer program code for determining the order in which said visual objects are to be placed based on said comparison; and computer program code for re-ordering and placing the visual objects in said overlapping condition in accordance with said determination.

[0017] According to still yet another aspect of the present invention there is provided a computer readable medium including a computer program for creating a connecting link between source and destination visual objects, said computer program comprising: computer program code for determining a region within each visual object to be joined and a connecting path extending between the regions; computer program code for clipping the connecting path so that the connecting path terminates at the locations where the connecting path intersects the source and destination visual objects; and computer program code for extending a connecting link between the source and destination visual objects terminating at said locations.

[0018] According to still yet another aspect of the present invention there is provided a computer readable medium including a computer program for creating a connecting link joining source and destination visual objects, said computer program comprising: computer program code for determining a path for said connecting link between said source and destination visual objects; and computer program code for representing said connecting link by at least one non-line shape.

[0019] The present invention provides advantages in that diagramming restrictions associated with conventional diagramming computer software programs are removed. As a result, the present invention provides diagramming tools that offer users increased freedom in how visual objects and connecting links interact. The overlapping object tool automatically places visual objects in an aesthetically pleasing way, without requiring tedious manual placement and/or adjustment of the visual objects. The object-connecting tool allows connecting links to be drawn between visual objects of virtually any shape and allows users to shape the connecting links to

provide enhanced visual meaning to the connecting links. When the overlapping object and object-connecting tools are integrated and used in conjunction, users are able to create diagrams with significantly increased visual richness.

Brief Description of the Drawings

[0020] Embodiments of the present invention will now be described more fully with reference to the accompanying drawings in which:

Figure 1 shows a set of visual objects presented on a computer display and arranged in a non-overlapping condition;

Figure 2 shows the set of visual objects of Figure 1 arranged in an overlapping condition using an overlapping object tool in accordance with the present invention;

Figure 3 shows another set of visual objects arranged in an overlapping condition using the overlapping object tool;

Figure 4a shows two visual objects joined by a connecting link that extends across an unrelated visual object;

Figure 4b shows the connecting link of Figure 4a placed behind the unrelated visual object by the object overlapping tool in accordance with a user customization;

Figure 5 shows two visual objects joined by a connecting link that extends partially across an unrelated visual object;

Figure 6 shows a source visual object and a destination visual object joined by a connecting link;

Figure 7 shows source and destination visual objects joined by connecting links of different path type;

Figures 8a to 8c show the steps performed during creation of a straight line connecting link joining source and destination visual objects;

Figure 9 shows the path of a self-looping connecting link;

Figure 10 shows one embodiment of link clipping of a self-looping connecting link;

Figure 11 shows another embodiment of linking clipping of a self-looping connecting link;

Figures 12a to 12d show examples of connecting links of different path type joining differently shaped visual objects;

Figures 13a and 13b show a prior art connecting link joining visual objects before and after manipulation of the visual objects;

Figures 14a and 14b show a connecting link joining visual objects created by the present object-connecting tool before and after manipulation of the visual objects;

Figure 15a shows examples of visual objects representing concepts;

Figure 15b shows examples of prior art connectors joining visual objects in concept maps;

Figure 16 shows a plurality of shapes for use with shaped connecting links;

Figures 17a and 17b show examples of shaped connecting links joining visual objects;

Figure 18 shows the determination of start and end points of a shaped connecting link;

Figure 19 shows the point along the connecting path of the shaped connecting link where the first shape is to be placed when the connecting path forms an angle between 0 and 90° with respect to the horizontal;

Figure 20 shows the point along the connecting path of the shaped connecting link where the first shape is to be placed when the connecting path forms an angle between 90° and 180° with respect to the horizontal;

Figure 21 shows the point along the connecting path of the shaped connecting link where the first shape is to be placed when the connecting path forms an angle between 180° and 270° with respect to the horizontal;

Figure 22 shows the point along the connecting path of the shaped connecting link where the first shape is to be placed when the connecting path forms an angle between 270° and 360° with respect to the horizontal;

Figure 23 shows an example of a shaped connecting link forming an angle with the horizontal and including shapes that have been rotated corresponding to the formed angle;

Figure 24 shows a shape at various rotations;

Figure 25 shows the point of a shape used to locate the shape along the connecting path;

Figure 26 shows a series of shapes painted along the connecting path and forming a shaped connecting link;

Figure 27 shows the situation when the distance between the start and end points of a straight line segment are less than the start location for the first shape;

Figure 28 shows the situation when the distance between the start and end points of a straight line segment is greater than or equal to start location for the first shape;

Figure 29 shows multiple shape locations along a straight line segment;

Figure 30 also shows multiple spaced shape locations along a straight line segment;

Figure 31 shows a curved shaped connecting link;

Figure 32 shows a self-looping shaped connecting link;

Figures 33 to 35 show different examples of visual objects joined by shaped connecting links;

Figures 36 and 37 show differently shaped and colored shapes for use in shaped connecting links;

Figure 38 shows two visual objects joined by a shaped connecting link including shapes rotated at different angles;

Figures 39 and 40 show a concept map representing regulation of water temperature including visual objects joined by shaped connecting links;

Figure 41 shows an automatic shape reorientation process; and

Figures 42 to 45 show examples of concept maps including visual objects joined by connecting links before and after being modified by the overlapping object and object-connecting tools.

Detailed Description of the Embodiments

[0021] The present invention provides diagramming tools that allow a user to create enhanced diagrams of visual objects. During creation of a diagram that includes visual objects, overlapping visual objects are automatically arranged in order of appearance to provide a three-dimensional context to the diagram. A wide variety

of shapes and representations of connecting links are available to link visual objects. In this manner, the user can connect visual objects in an appealing manner and that provides a more meaningful representation.

[0022] The present invention can be incorporated into a wide range of computer application programs that manage the use of visual objects thereby to provide enhanced functionality to those computer application programs. These types of computer application programs include, but are not limited to concept mapping, flow-charting, mind mapping, technical engineering, diagramming, white boarding, gaming, visualization tools, graphical layout and design, three-dimensional modeling and animation programs.

[0023] The present diagramming tools include an overlapping object tool and an object-connecting tool. The overlapping object tool automatically overlaps displayed visual objects to provide z-order to the displayed visual objects without requiring the user to navigate through displayed menus as is required in the prior art. The automatic arrangement of visual objects places overlapping visual objects in a favourable way to what the user expects and enables the user to arrange visual objects without accidentally moving a visual object out-of-sight. The object-connecting tool allows connecting links to be drawn between visual objects of virtually any shape. The object-connecting tool also determines the shortest distance for connecting links interconnecting arbitrarily shaped visual objects and allows connecting links to be shaped providing enhanced visual meaning to the connecting links. The overlapping object tool and the object-connecting tool can be run independently or in conjunction depending on the needs of the user. Further specifies of the overlapping object tool and object-connecting tool will now be described with reference to Figures 1 to 45.

Overlapping Object Tool

[0024] The overlapping object tool compares displayed visual objects to determine the relative sizes of the visual objects to be placed in an overlapping condition and automatically overlaps the visual objects i.e. changes the z-order of the overlapping visual objects. During the overlapping process, larger visual objects are automatically placed behind smaller visual objects. The operation of the overlapping object tool will now be described with reference to Figures 1 to 5.

[0025] Turning now to Figure 1, a display is shown presenting a set of two-dimensional, differently shaped, non-overlapping visual objects VO. If the visual objects are to be placed in an overlapping condition, the overlapping object tool is run. When run, the overlapping object tool compares each of the displayed visual objects VO to determine the relative sizes of the visual objects i.e. which visual objects are bigger than others. The overlapping object tool then automatically arranges the visual objects so that they overlap with bigger visual objects appearing behind smaller visual objects as shown in Figure 2. The pseudo code below represents the above operation:

Automatic Overlap Visual Object A amongst a set C:
-for each visual object B in set C in order of overlap, from the bottom to the top
-if visual object A is bigger than visual object B, insert visual object A behind visual object B

[0026] In order to determine whether a visual object is bigger than another visual object, in the present embodiment, the overlapping object tool calculates the areas of rectangular regions surrounding the visual objects and compares the calculated areas. Calculating the areas of the rectangular regions can be performed quickly and easily. Comparing the calculated areas typically yields results that properly overlap the visual objects.

[0027] If desired, other measures can be used to determine the relative sizes of the visual objects. For example, rather than calculating the areas of rectangular regions surrounding the visual objects, the overlapping object tool can calculate the actual display area taken up by the visual objects using a number of techniques such as for example polygon approximations or viewable pixel summations.

[0028] As will be appreciated depending on the nature of the visual objects, the measure used to determine whether a visual object is "bigger" than another visual object may vary. If the visual objects are three-dimensional, a comparison of volume or other dimension or prominent metric may be made to determine whether a visual object is bigger than another visual object. If the visual objects are heterogeneous, i.e., different shapes and forms, the measure selected to compare the visual objects should apply to all of the visual objects to be placed in an overlapping condition so that the desired z-order of the visual objects can be achieved. Figure 3 shows a number of

heterogeneous visual objects placed in an overlapping condition by the overlapping object tool. As can seen, larger visual objects appear behind smaller visual objects.

[0029] The overlapping object tool is preferably used with a computer application program that allows visual objects to be manipulated on-screen and automatically changes the z-order of visual objects as visual objects are dragged or moved into an overlapping condition thereby to provide real-time visual object re-ordering. Of course, if desired, the overlapping object tool can be conditioned to change the z-order of a visual object brought into an overlapping condition only after the visual object has been released from the drag or move operation. Alternatively, the overlapping object tool can be conditioned to change the z-order of overlapping visual objects in response to an input command. In this manner, a user is able to drag or move one or more visual objects into an overlapping condition and then subsequently run the overlapping object tool to re-order the overlapping visual objects.

[0030] The overlapping object tool allows a user to customize the visual object ordering rules so that symbols such as connecting links relating to and linking two or more visual objects are handled in accordance with the user's preferences. For example, if a connecting link CL joining two visual objects completely crosses an unrelated visual object VO_{UR} as shown in Figure 4a, the overlapping object tool can be conditioned to place the connecting link beneath the unrelated visual object VO_{UR} as shown in Figure 4b. If a connecting link CL intersects an unrelated visual object VO_{UR} but terminates within that unrelated visual object, the overlapping object tool maintains the connecting link CL above the unrelated visual object VO_{UR} as shown in Figure 5. Other visual object ordering rule customizations are available to the user to deal with other connecting link conditions such as for example when a connecting link joins two overlapping visual objects and one end of the connecting link is obscured by the visual object that overlaps the other visual object.

Object-Connecting Tool

[0031] Unlike prior art diagramming tools, the present object-connecting tool avoids the use of rigidly defined anchor points for terminating connecting links on visual objects. As a result, the orientation of curved connecting links extending between visual objects is maintained as the visual objects are manipulated. Also, the object-connecting tool joins visual objects with connecting links of the shortest possible

length. Since anchor points are not used to terminate visual objects, visual objects can take on virtually any shape and still be joined by connecting links that fully extend between the visual objects. As a result, connecting links can be used to join visual objects that include transparent regions.

[0032] The object-connecting tool defines a connecting link as a path that joins a source visual object and a destination visual object as shown in Figure 6. Each of the source and destination visual objects is a node object that either has a shape or is a finite point. At each end of the connecting link there can either be a zero or an arrowhead pointing to the visual object.

[0033] The path of the connecting link is abstractly defined and includes a designated path type selected from the group of a straight line, a cubic curve, a right-angled line or a circular self-loop as shown in Figure 7. The path also defines properties of the connecting link. The path properties include pattern, link shape, thickness, color and control points. The pattern property determines whether the connecting link is solid, dotted, dashed, etc. The link shape determines whether the connecting link is represented by a line or by a series of non-line shapes as will be described.

[0034] During creation of a connecting link extending between source and destination visual objects, the object-connecting tool determines a connecting path that joins the centers of the source and destination visual objects. The object-connecting tool then clips the connecting path using a link clipping process to determine accurately the intersection of the connecting path with the boundaries of the source and destination visual objects so that the connecting link terminates at the intersection points. If appropriate arrowheads are then added to the end or ends of the connecting link.

[0035] During the link clipping process, the object-connecting tool flattens the connecting link into line segments, if necessary i.e. if the connecting link is curved, and then calls a **contains()** method that queries the source and destination visual objects for information concerning their shapes. Visual objects include information concerning the bounds of their shapes. During a query by the **contains()** method, two-dimensional points along the connecting path are examined to determine whether the points lie within a visual object. The **contains()** method query returns a true result if a point on the connecting path lies within a visual object and a false result otherwise.

This allows the point along the connecting path where the connecting path initially intersects each visual object to be determined and hence the points along the connecting path where the connecting path is to be clipped.

[0036] As mentioned above, the object-connecting tool supports irregular shaped visual objects including those having transparent regions. To avoid a connecting path being clipped within a transparent region, the **contains()** method does not consider transparent regions within a visual object to be part of the visual object for the purpose of link clipping. To support such a visual object, a transparency bitmap is included defining transparent regions of the visual object.

[0037] Following link clipping, the object-connecting tool places an arrowhead at the end or ends of the connecting link if appropriate. The orientation of each arrowhead depends on the locations of the source and destination visual objects relative to one another and the path of the connecting link. In order to compute arrowhead orientation, the destination vectors of the source and destination visual objects are determined using the connecting link or flattened line segments if the connecting link is curved. Once the arrowhead orientation is determined, each arrowhead is placed at the end of the connecting link with the tip of the arrowhead terminating at the visual object. The clip-point along the connecting path is then translated by the length of the arrowhead in the direction opposite that of the destination vector so that the connecting link ends at the backend of the arrowhead.

[0038] Turning now to Figures 8a to 8c, the above-described connecting link creation process performed by the object-connecting tool for a straight line connecting link joining source and destination visual objects is shown. Initially, a straight line connecting path CL joining the centers of the source and destination visual objects is determined as shown in Figure 8a. Once determined, the link clipping process is invoked. In this example, since the connecting path CL is a straight line in the form a single line segment, no flattening of the connecting path is required. During link clipping, the object-connecting tool calls the **contains()** method which traverses the connecting path CL to determine the precise points along the connecting path where the connecting path intersects the source and destination visual objects and hence the clip-points for the connecting path. Once the clip-points have been determined, the connecting path is clipped resulting in a connecting link extending between the visual objects. In this example, an arrowhead is placed at one end of the connecting link as

shown in Figure 8b. The orientation of the arrowhead is computed using the destination vector. With the arrowhead placed at the end of the connecting link, the clip-point is translated by the length of the arrowhead, in the direction opposite that of the destination vector so that the connecting link ends at the backend of the arrowhead as shown in Figure 8c.

[0039] As mentioned above with reference to Figure 7, a curved connecting link extending between source and destination visual objects is represented by a cubic curve. The centers of the source and destination visual objects and a control point that can be manipulated by a user, define the cubic curve. During link clipping of a curved connecting link, the link path is flattened into a poly-line approximation of the cubic curve including a plurality of straight line segments. During the **contains()** method, each straight line segment is traversed to determine the clip-points for the connecting path.

[0040] The link path of a self-looping connecting link has the same visual object as its source and destination. A circle having a diameter formed by a line segment joining the center of the visual object and an external control point defines the path of the self-looping connecting link as shown in Figure 9. During link clipping of a self-looping connecting path, Bresenham's circle drawing algorithm is used to compute the point array defining a flattened circular path of the connecting path as shown in Figure 10. The point array is then traversed in both clockwise and anti-clockwise directions to determine the clip-points along the connecting path. If arrowheads are to be placed at the ends of the self-looping connecting link, the point array is then traversed in the opposite direction from each clip-point, until the distance from the clip-point is equal to the length of the arrowhead. The arrowheads are then placed at the ends of the connecting link with their orientation being determined by the tangent vectors to the circle at the end points of the self-looping connecting link.

[0041] Alternatively, a binary search method may be used to determine the clip-points along the self-looping connecting path as shown in Figure 11. Although this method works, the results are not ideal if the visual object is not convex-shaped. Also, when using the binary search method the arrowheads may get distorted if the diameter of the circle defining the self-looping connecting path is small.

[0042] As will be appreciated, the object-connecting tool allows connecting links to be properly clipped so that they terminate at the intersection points of the source

and destination visual objects. Since the object-connecting tool does not rely on anchor points to terminate connecting links, connecting links terminate at the visual objects regardless of the shapes of the visual objects. Figures 12a to 12d show examples of connecting links of different path type joining differently shaped visual objects. As will be appreciated, the connecting links terminate at the visual objects. Also, since connecting links are determined from connecting paths joining the centers of the visual objects, the shapes of the connecting links are preserved after visual objects are manipulated. Figures 13a and 13b show a prior art connecting link joining two visual objects before and after manipulation. Since the prior art connecting link is joined to the visual objects at anchor points, the shape of the connecting link becomes distorted after the visual objects are manipulated (Figure 13b). Figures 14a and 14b show a connecting link joining two visual objects created by the present object-connecting tool before and after manipulation. Since the connecting link is not anchored to the visual objects, the shape of the connecting link is preserved after the visual objects are manipulated.

[0043] Although specific connecting link path types have been described above, it will be appreciated that virtually any connecting link path type can be used. For example bezier curves, poly-lines or even compound paths can be used to join visual objects. During link clipping, these connecting paths are flattened into straight line segments that are traversed to determine the clip-points.

[0044] In addition to connecting visual objects with connecting links in the form of straight, dashed or dotted lines, the object-connecting tool also allows connecting links to be represented by non-line shapes extending along the lengths of the connecting paths. This gives the connecting links more meaning than is available when using lines as the connecting links. Providing this additional meaning has advantages particularly in the case of concept maps. As is known, a concept map is a diagram used to explore, share and teach information that includes visual objects representing concepts and connectors joining the visual objects. In the past, although many different styles of visual objects have been used in such concepts maps, the types of connectors used have been limited. In fact to-date, the only variations in connectors have been line shape i.e. straight, right-angled or curved, line color and line pattern i.e. solid, dashed or dotted. In all cases the connectors have been lines. Figure 15a shows examples of visual objects representing concepts and Figure 15b

shows examples of prior art line connectors joining visual objects. Without text accompanying these line connectors, it is difficult to determine the relationship between the joined visual objects. Since the present object-connecting tool allows connecting links to be represented by shapes, in the case of concept maps, the connecting links allow concept maps to be more quickly understood and provide more information to a viewer at a glance. The connecting links visually convey the relationship between the visual objects that they join.

[0045] Shaped connecting links generated by the object-connecting tool typically include one or more shapes drawn repeatedly at generally evenly spaced locations along their lengths. The object-connecting tool can be conditioned to apply the shapes associated with the shaped connecting link as the shaped connecting link is being drawn or can be conditioned to draw the shapes after the connecting link has been created. The object-connecting tool also allows the shapes associated with a shaped connecting link to be changed. The shapes used with shaped connecting links provide a pictorial representation of a thing, an arbitrary or conventional sign used to represent operations, quantities, elements, relations or qualities, or a geometric object. For example, the shapes may include light bulbs, foot or animal prints, emoticons such as happy or sad faces, paper clips, lightening bolts, money symbols or other symbol, shape or graphic that provides a meaning. Figure 16 shows a plurality of possible shapes for use with shaped connecting links. Figure 17a is similar to Figure 15 but shows visual objects joined by shaped connecting links. Figure 17b also shows visual objects joined by shaped connecting links.

[0046] Shapes used by the object-connecting tool during creation of a shaped connecting link are stored in memory. The object-connecting tool in the present embodiment includes a toolkit storing a plurality of predefined shapes from which a user may select. The toolkit of the object-connecting tool also allows a user to create shapes. During shape creation, the outline co-ordinates and color of the shape are defined by the user. The size of the shape is then set. All areas associated with the shape are then created. For example, a shape in the form of a duck could have the area specified by the following outline co-ordinates of:

$$x = \{1, 4, 7, 9, 4, 6, 3, 5, 7\}, y = \{3,4,5,3,4,5,6,7,8\}.$$

[0047] Shapes can also be created from the predefined shapes supplied by the toolkit of the object-connecting tool. For example, a happy face can be created using a circle to define the outline of the happy face and ellipses to form the eyes. Shapes can also be created by specifying outline co-ordinates and using predefined shapes supplied by the toolkit of the object-connecting tool. Each shape includes an orientation vector that defines its upright orientation.

[0048] When it is desired to generate a shaped connecting link, the shape to be used to represent the connecting link is selected. Once the shape has been selected, the object-connecting tool uses a path mapping technique to determine the co-ordinates along the path of the connecting link where the shapes are to be painted. The path mapping technique varies depending on the shape of the connecting link used to join the visual objects.

[0049] If the shapes are to be painted along a shaped connecting link that follows a straight line, the start and end points of the shaped connecting link are firstly determined as shown in Figure 18.

[0050] Next, a spacing distance between the centres of adjacent shapes is set to the width of the shape plus a few extra pixels to provide a desired spacing between the shapes. The angle of the slope of the connecting link is then determined using the following formula:

$$\theta = \tan^{-1}\left(\frac{y_{p2} - y_{p3}}{x_{p3} - x_{p2}}\right)$$

where (x_{p2}, y_{p2}) and (x_{p3}, y_{P3}) are the co-ordinates of p2 and p3 respectively. The values y_{p2} and y_{P3} are reversed to compensate for the fact that the y cartesian values decrease from top to bottom. Using this angle, the changes in x and y corresponding to a shift along the connecting link equal to the spacing distance are determined as follows:

$$x_{SD} = abs\ (spacing\ distance \cdot cos\theta)$$

$$y_{SD} = abs\ (spacing\ distance \cdot sin\theta)$$

[0051] The co-ordinates x_1Pt and y_1Pt along the shaped connecting link where the first shape is to be painted are then determined. For ease of illustration, it will be assumed that the shaped connecting link is sufficiently long enough to accommodate

at least one shape. In some circumstances, it can be desired to place the first shape along the connecting link a distance different than the spacing distance between adjacent shapes, but for the purpose of simplicity, it will be assumed for this straight connecting link that the starting distance is equal to the spacing distance. In this example, it will be assumed that the connecting link spans a height equal to 15 units and a width equal to 9 units. There are four cases that determine the co-ordinates along the connecting link where the first shape will be painted dependent on the angle of the straight connecting link.

[0052] If the connecting link forms an angle between 0 and 90 degrees with respect to the horizontal, the position of the first shape along the connecting link is determined as follows. The first shape is spaced from the tail end of the connecting link, P_2, by a distance equal to the spacing distance to be used between shapes as shown in Figure 19. Subsequent shapes are spaced along the connecting link in a similar manner. The object-connecting tool determines the co-ordinates for each shape in the manner identified below:

Co-ordinates: p2 = (100, 120) p3 = (109, 105)
p2.x becomes small_x; (since p2.x < p3.x)
p2.y becomes large_y; (since p2.y > p3.y)

Thus,

$x_iPt = small\_x + i \cdot x_{SD}$; *and*
$y_iPt = large\_y - i \cdot y_{SD}$

where i is the number of the shape for which the co-ordinates are being determined.

[0053] If the connecting link forms an angle between 90 and 180 degrees with respect to the horizontal, the first shape is drawn along the connecting link at the position shown by the asterisk in Figure 20. The object-connecting tool determines the co-ordinates for each shape in the manner identified below:

Co-ordinates: p2 = (109, 120) p3 = (100, 105)
p2.x becomes large_x; (since p2.x > p3.x)
p2.y becomes large_y; (since p2.y > p3.y)

Thus,

x_iPt = large_x - $i \cdot x_{SD}$; and
y_iPT = large_y - $i \cdot y_{SD}$

[0054] If the connecting link forms an angle between 180 and 270 degrees with respect to the horizontal, the first shape is drawn along the connecting link at the position shown by the asterisk in Figure 21. The object-connecting tool determines the co-ordinates for each shape in the manner identified below:

Co-ordinates: p2 = (109, 105) p3 = (100, 120)
p2.x becomes large_x; (since p2.x > p3.x)
p2.y becomes small_y; (since p2.y < p3.y)

Thus,

x_iPt = large_x - $i \cdot x_{SD}$; and
y_iPt = small_y + $i \cdot y_{SD}$

[0055] If the connecting link forms an angle between 270 and 360 degrees with respect to the horizontal, the first shape is drawn along the connecting link at the position shown by the asterisk in Figure 22. The object-connecting tool determines the co-ordinates for each shape in the manner identified below:

Co-ordinates: p2 = (100, 105) p3 = (109, 120)
p2.x becomes small_x; (since p2.x < p3.x)
p2.y becomes small_y; (since p2.y < p3.y)

Thus,

x_iPt = small_x + $i \cdot x_{SD}$; and
y_iPt = small_y + $i \cdot y_{SD}$

[0056] Once the co-ordinates for the shapes along the path of the shaped connecting link are determined, the rotation angle of the shapes to be painted is

determined based on the angle formed between the connecting link and the horizontal. The rotation is determined using the following:

If (θ mod 180°) < 90°,

rotation = counter-clockwise (θ mod 180°)

If (θ mod 180°) $\geq$ 90°,

rotation = clockwise [180° - (θ mod 180°)]

While this method of determining the rotation is used to provide shapes that are right side-up, it can be desirable in some circumstances to have upside-down shapes.

[0057] The rotation angle is then used to orient the shapes as they are painted. Figure 23 shows an example of a shaped connecting link forming an angle with the horizontal and including shapes that have been rotated corresponding to the formed angle. Figure 24 shows a shape at various rotations.

[0058] With the shape co-ordinates and shape rotation determined, the first shape is painted along the path of the connecting link. During painting the object-connecting tool paints the shape so that the center of the shape is positioned at the determined shape co-ordinates. To achieve this, the object-connecting tool specifies where the top left corner of the shape is to be placed as shown in Figure 25. For the shape to be centered at the shape co-ordinates, the co-ordinates of the top left corner of the shape (prior to rotation) are calculated by subtracting one half of the height of the shape from the y-co-ordinate and subtracting one half of the width of the shape from the x-co-ordinate. These co-ordinates are then used to paint the shape.

[0059] With the first shape painted, the co-ordinates along the path of the connecting line where the next shape is to be painted are determined by translating along the path a distance equal to the shape spacing distance. This is performed by using the appropriate formula depending on the angle of the connecting link and with the appropriate value for i corresponding to the number of the shape. As the connecting link in this example is a straight line, the orientation of each subsequent shape is equal to that of the first. The second shape is then painted at the position (x_2Pt, y_2Pt). This process continues until the distance between the end of the path of the connecting link and the last painted shape is less than the shape spacing distance as shown in Figure 26.

[0060] Creating a curved shaped connecting link is similar to creating a straight line shaped connecting link, however there are some differences. As mentioned above, the curved connecting link is flattened into a series of straight line segments.

[0061] Assuming each shape is to be spaced an equal distance apart, the spacing distance is set to the width of the shape plus a few extra pixels to provide the desired space between shapes. The co-ordinates of the start and end points of the first straight line segment (i.e., p_1 and p_2) from the set that makes up the curved connecting link are first obtained. The distance between the two points is then compared with a starting distance that is set to be equal to the shape spacing distance.

[0062] If the distance between the start and end points of the straight line segment in question is less than the starting distance, such as shown in Figure 27, there is not enough room to paint the shape on the straight line segment. This distance between the two end co-ordinates is then added to a variable *carryover*, which is initialized to zero at the start for a curved connecting link.

[0063] If the distance between the start and end points of the straight line segment is greater than or equal to the starting distance, such as shown in Figure 28, the positions of shapes to be painted therealong are determined. Figure 29 shows how multiple spacing distances may be placed between points p1 and p2. As a result, multiple shapes may be placed between points p1 and p2. The same process used to paint shapes on a straight line connecting link, as described above, is used to paint one or more shapes along the first straight line segment of the curved connector.

[0064] The distance along the straight line segment between the last shape and the end of the straight line segment as shown in Figure 30 is then assigned to the variable *carryover*.

[0065] After the first straight line segment of the curved connecting link has been addressed, the next straight line segment along the curved connecting link is examined. As it is desired to space the shapes equally along the entire length of the curved connecting link, if a shape was not painted at the end of the previous straight line segment, the first shape can be placed closer to the start of the straight line segment being examined. The starting distance representing the placement of the first shape along the straight line segment being examined is reduced by the variable *carryover*, the amount of unused length of the previous straight line section:

starting distance = *spacing distance* - carryover

[0066] If the starting distance is greater than the length of the straight line segment being examined, the variable *carryover* is adjusted to reflect the increased distance between the last shape along the curved connecting link:

carryover = carryover + *length of straight line section*

[0067] If the starting distance is less than or equal to the length of the straight line segment being examined, a shape is painted at a distance along the straight line segment corresponding to the starting distance. Subsequent shapes are painted along the length of the straight line segment, each separated by the spacing distance. The co-ordinates for these shapes are determined in accordance with the above described formulas for the different shapes, but adjusted as follows:

For straight line segments at an angle between 0 and 90 degrees from the horizontal:

$$x_iPt = small\_x + \left(\frac{\text{starting distance}}{\text{spacing distance}} + i\right)x_{SD}$$

$$y_iPt = large\_y - \left(\frac{\text{starting distance}}{\text{spacing distance}} + i\right)y_{SD}$$

Any remaining length of the straight line segment shorter than the spacing distance is then assigned to the variable *carryover*, replacing its current value. If the straight line segment being examined is the last of the curved connecting link, any *carryover* can be discarded. Figure 31 shows an exemplary curved connecting link generated in accordance with the above-described process.

[0068] Shaped self-looping connecting links can also be created. Figure 32 illustrates the start and end points of a shaped self-looping connector link. In this case, a modified approach is followed during painting of the shapes. Firstly, the number of shapes to be painted along the shaped self-looping connecting link is determined as follows:

$$numberofshapes = \frac{\text{circumference}}{\text{spacing distance}}$$

where the number of shapes is rounded down if it is not an integer value.

[0069] The origin (x_0,y_0) of the circle defining the shaped self-looping connecting link is then determined. This can be done using a number of techniques, including locating of the mid-value in the range of x and y values. Next, the shapes are painted along the circumference of the circle at positions determined using the following formulas.

$$x_i = x_0 + radius \bullet \cos\left(i \bullet \frac{360^\circ}{numberofshapes} \right)$$

$$y_i = y_0 + radius \bullet \sin\left(i \bullet \frac{360^\circ}{numberofshapes} \right)$$

[0070] Turning now to Figures 33 to 35, examples of visual objects joined by shaped connecting links are illustrated. For example, Figure 33 shows shaped connecting links joining visual objects representing buildings. In this case, the shapes used along the connecting links represent paths extending between the buildings and provide an indication as to whether the paths have been cleared of snow. Figure 34 shows two visual objects joined by a shaped connecting link including different shapes along the shaped connecting link.

[0071] Figure 35 shows two visual objects joined by a shaped connecting link including a single shape that has been stretched to extend fully between the visual objects. In this particular example, the stretched shape is a dog, with the head and tail of the dog showing direction as an alternative to conventional arrowheads.

[0072] The shapes used along shaped connecting links may vary in size and/or color as shown in Figures 36 and 37.

[0073] Figure 38 shows two visual objects joined by a shaped connecting link including spaced light bulbs. In this example, the light bulbs are rotated at different angles along the shaped connecting link depending on their locations.

[0074] Figures 39 and 40 show a concept map representing regulation of water temperature in a swimming pool including visual objects joined by shaped connecting links. In this case, the shapes along the shaped connecting link change depending on the state of the temperature regulation process. In Figure 39, sad faces are used along the shaped connecting link when the temperature of water in the swimming pool

is below the desired temperature. In Figure 40b, happy faces are used along the shaped connecting link when the temperature of water in the swimming pool is at the desired temperature.

[0075] If desired, the object-connecting tool can be conditioned to reorient automatically the shapes used along shaped connecting link in accordance with their orientation vectors. In this case, if visual objects are dragged or moved causing the shapes along shaped connecting links to become inverted, the object-connecting tool automatically re-orients the shapes to return them to the desired orientation. The object-connecting tool determines that a shape is inverted when its orientation vector rotates past the horizontal. To reorient the shape, the object-connecting tool either rotates the shape by 180 degrees or remaps the vertical drawing co-ordinates of the shape to negative values. Figure 41 shows the automatic shape reorientation process. As can be seen, visual objects joined by a shaped connecting link are moved relative to one another to a condition where the shapes along the shaped connecting link are inverted. The shapes along the shaped connecting link after having been reinverted by the object-connecting tool.

[0076] If desired, sounds can be associated with the shaped connecting links. For example, a continuous sound may be played when a shaped connecting link is drawn. Alternatively, a sound may be played when a shape along a shaped connecting link is selected via a mouse click or by other operation.

[0077] If desired, the shapes can be animated with the animation being presented continuously or in response to an event. For example, when a visual object is moved, the shapes along the connecting path can animate the result. Alternatively, interactions may be associated with the shapes along the connecting path. For example, when a shape is selected a tool tip elaborating on the connection between the visual objects may be displayed.

Integrated Overlapping Object and Object-Connecting Tools

[0078] The above-described overlapping object tool and object-connecting tool can be integrated to provide enhanced functionality. In particular when integrated, the overlapping object and object-connecting tools provide for the automatic positioning of overlapping visual objects and the precise connection of visual objects with connecting links whether shaped or not. Figures 42 to 45 show examples of concept maps

including visual objects joined by connecting links before and after being modified by the integrated overlapping object and object-connecting tools. In Figure 42, the connecting link extending between the visual objects is extended by the object-connecting tool so that the connecting link extends fully between the visual objects. In Figure 43, the visual objects and shaped connecting link are automatically re-positioned by the overlapping object tool. In Figure 44, visual objects are automatically placed in an overlapping condition and re-positioned by the overlapping object tool.

[0079] Figure 45 shows a concept map created using the integrated overlapping object and object-connecting tools. As can be seen, the visual objects representing food have been automatically placed in an overlapping condition by the overlapping object tool. A shaped connecting link extends fully between the visual objects representing food and the visual object representing an ant. The shapes along the shaped connecting link are in the form of small ants representing the crawling action of the visual object representing the ant to the visual objects representing food.

[0080] As will be appreciated, the overlapping object and object-connecting tools when integrated allow arbitrarily shaped visual objects to be automatically laid out on top of one another and joined with connecting links that extend fully between the visual objects. The connecting links can be in the form of lines of shortest distance and including one or more arrowheads. The connecting links may also be shaped to provide semantic meaning concerning the connection between the visual objects.

[0081] The present invention can be embodied as computer readable program code stored on a computer readable medium. The computer readable medium is any data storage device that can store data, which can thereafter be read by a computer system. Examples of computer readable medium include read-only memory, random-access memory, CD-ROMs, magnetic tape and optical data storage devices. The computer readable program code can also be distributed over a network including coupled computer systems so that the computer readable program code is stored and executed in a distributed fashion.

[0082] Although a number of embodiments have been described above, those of skill in the art will appreciate that variations and modifications may be made without departing from the spirit and scope thereof as defined by the appended claims.